J. Symbolic Computation (1987) 3, 203–216

Complexity of Matching Problems

DAN BENANAV, DEEPAK KAPUR AND
PALIATH NARENDRAN

Corporate Research and Development
General Electric Company
Schenectady, NY

The associative-commutative matching problem is shown to be NP-complete; more precisely, the matching problem for terms in which some function symbols are uninterpreted and others are both associative and commutative, is NP-complete. It turns out that the similar problems of associative-matching and commutative-matching are also NP-complete. However, if every variable appears at most once in a term being matched, then the associative-commutative matching problem is shown to have an upper-bound of $O(|s| * |t|^3)$, where $|s|$ and $|t|$ are respectively the sizes of the pattern s and the subject t.

1. Introduction

Matching and unification algorithms play an important role in such areas as logic programming, functional and relational programming, automated reasoning, program verification and specification analysis. Several papers have appeared that investigate the complexity of general term matching and related problems (Dwork et al. (1986), Hoffman and O'Donnell (1982), Overmars and Van Leeuwen (1979), Steyaert and Flajolet (1983)). Here we consider specialized instances of these problems, namely when certain operators under consideration have the associative and commutative properties. To our knowledge, Plotkin (1972) was the first one to have studied an approach for handling such operators by integrating them into unification algorithms. Since then, methods have

0747–7171/87/010203+14 $03.00/0

been suggested for handling such operators in resolution proof systems using the paramodulation and narrowing techniques (Lankford (1975), Lankford and Ballantyne (1979), Slagle (1974)) and in term rewriting systems by suitably extending the Knuth-Bendix (1970) completion procedure (Lankford and Ballantyne (1977), Peterson and Stickel (1981)). In both these applications, associative-commutative matching is extensively performed. Furthermore, from various computer implementations of these applications, especially those based on rewriting techniques, it is clear that the performance of associative-commutative matching algorithms dominates the overall performance (Hullot (1979), Kapur et al (1986)).

The associative-commutative matching problem is shown to be NP-complete; this result is an indication why a general associative-matching algorithm is not likely to perform well. We show that in fact, commutative-matching as well as associative-matching are also NP-complete. However, for a restricted version of associative-commutative matching, where every variable in a term being matched has a unique occurrence, the problem is shown to have an upper bound of $O(|s| * |t|^3)$, where $|s|$ and $|t|$ are respectively the sizes of the pattern s and the subject t.

The constructions employed in showing these NP-completeness results are quite similar to one another. The 3SAT problem (satisfiability problem for the case when every clause has 3 literals) and one of its variants (Monotone 3SAT) are used in the reductions. There appear to be three steps in the construction: (i) simulating boolean values 'true' and 'false' using ground terms, (ii) constructing terms from a 3SAT formula in such a way that (a) variables of the formula can get either of the boolean values, and (b) there is a match if and only if that match, when suitably interpreted, is a satisfying assignment for the formula, and (iii) extra variables are used to match the subterms that are 'left over', namely those not used for matching variables in the formula.

Section 2 contains some basic definitions. In Section 3, various problems referred to in the paper are defined. Section 4 shows that associative-commutative matching problem is NP-complete. In Sec-

tion 5, we show that if every variable in a term being matched appears uniquely, then the associative-commutative matching problem has a polynomial upper-bound. The NP-completeness of the associative-matching problem and the commutative-matching problem are proved respectively in Sections 6 and 7.

2. Definitions

Let F be a finite set of function symbols of fixed arity and X be a denumerable set of variables. By $T(F, X)$ we denote the set of all possible terms that can be constructed using F and X. For a term t, $Var(t)$ denotes the set of all variables that occur in t. For example, $Var(f(x, y, g(y))) = \{x, y\}$. The *size* of a term s is the number of occurrences of function and variable symbols in s and is denoted by $|s|$.

A function f is *associative* if and only if it satisfies the following axiom:

$$f(f(x, y), z) = f(x, f(y, z)). \tag{A}$$

A function f is *commutative* if and only if it satisfies

$$f(x, y) = f(y, x). \tag{C}$$

We often refer to a function that is both associative and commutative as an 'ac-operator'.

A term t that involves associative function symbols is often represented in 'flattened' form. For example, if f is associative, then $f(a, f(b, c))$ is represented as $f(a, b, c)$. (In other words, f is treated as a varyadic symbol.) Flattening a term with respect to a function f can be done as follows: first represent a term in right-associative form. Such a term will be of the form $f(t_1, f(t_2, \cdots, f(t_{n-1}, t_n) \cdots)$ where $t_1, t_2, \ldots, t_n$ do not start with f. Then we simply represent the term as $f(t_1, t_2, \cdots, t_n)$. It can be easily shown that flattening a term can be done in linear time with respect to the size of the term.

A *substitution* is a mapping θ from variable names to terms such that $\theta(v) = v$ for all but a finite number of variable symbols. It can be denoted by an expression of the form $\{v_1 \leftarrow t_1, \cdots, v_k \leftarrow t_k\}$, where the $k \geq 0$ variable symbols $v_1, \cdots, v_k$ are distinct. (The case $k = 0$ is the identity substitution.) By the size of a substitution θ denoted in this way we shall mean $k + \sum_{i=1}^{k} |t_i|$. We shall denote the size of θ by $|\theta|$. The domain of a substitution θ is extended to the set of all terms by inductively defining $\theta(f(t_1, \cdots, t_n))$ to be $f(\theta(t_1), \cdots, \theta(t_n))$.

A substitution θ is said to *match* a term s with a term t if and only if $t = \theta(s)$. In this case, s is often called the *pattern* and t the *subject.*

Two terms s and t are said to be *associative-commutative equivalent*, expressed as $s \underset{ac}{=} t$, if and only if they are equivalent under the equational theory of the axioms (A) and (C). For example, if f is associative and commutative, then $f(f(a, b), c) \underset{ac}{=} f(c, f(b, a))$. We similarly define associative equivalence $(\underset{A}{=})$ and commutative equivalence $(\underset{C}{=})$.

3. Problem Definitions

The following is a list of problems that will be referred to in this paper.

1. **Associative-Commutative Equivalence** (referred to as **ACEQ**)

   Instance: A set of variable symbols V, a set of function symbols F some of which may be associative and commutative, and terms t_1, t_2 from $T(F, V)$.

   Question: Is $t_1 \underset{ac}{=} t_2$?

2. **Associative-Commutative Matching** (referred to as **ACM**)

   Instance: A set of variable symbols V, a set of function symbols F some of which may be associative and commutative, and terms t_1, t_2 from $T(F,V)$.

   Question: Does there exist a θ such that $\theta(t_1) \underset{ac}{=} t_2$?

3. **Distinct-Occurrences ACM** (referred to as **DO-ACM**)

   Instance: A set of variable symbols V, a set of function symbols F, some of which may be associative and commutative, two terms t_1, t_2 from $T(F,V)$ where every variable in t_1 occurs only once. (In other words, $v \in var(t_1) \Longrightarrow$ #occurrences of v in $t_1 = 1$).

   Question: Does there exist a θ such that $\theta(t_1) \underset{ac}{=} t_2$?

4. **Monotone 3SAT** (referred to as **Mono-3SAT**)

   Instance: A set of variables $U = \{z_1, z_2, \ldots, z_m\}$, and a set of clauses $C = \{c_1, c_2, \ldots, c_n\}$ such that
   1) each clause contains exactly three literals, and
   2) the literals in each clause are either all-positive or all-negative.

   Question: Does there exist a truth assignment $I: U \rightarrow \{1,0\}$ such that C is satisfiable ?

   This problem is known to be an NP-complete problem (Garey and Johnson, 1979).

4. Associative-Commutative Matching is NP-Complete

Theorem 1: ACM is NP complete.

Proof: It is easy to show that ACM $\in$ NP. Let t_1, t_2 be terms and θ be a substitution such that $\theta(t_1) \underset{ac}{=} t_2$. Clearly the size of θ cannot be greater than $(|t_2| + |t_1|)$. Thus given two terms t_1 and t_2 as input, we merely have to
(a) choose θ such that $|\theta| \leq (|t_1| + |t_2|)$,

(b) apply θ to t_1 to get a new term t_1', and

(c) check whether $t_1' \underset{ac}{=} t_2$.

Step (b) can obviously be done in polynomial time. Step (c) can also be done in polynomial time since ACEQ $\in$ P as will be shown later in the paper.

To show that ACM is NP-complete we will show that Mono-3SAT is polynomially transformable to ACM.

Suppose that we are given an instance of the Mono-3SAT problem with $U = \{z_1, z_2, \ldots, z_m\}$ and $C = \{c_1, c_2, \ldots, c_n\}$. Let f be an ac-operator, g an operator of arity n that is neither commutative nor associative and let 1 and 0 be nullary operators (constants). Let $U' = \{u_{11}, u_{12}, \ldots, u_{n1}, u_{n2}\}$ be a set of dummy variables (two per clause) with $U \cap U' = \emptyset$. Let $V = U \cup U'$ and $F = \{f, g, 1, 0\}$. We simulate the truth and falsity of a boolean variable z_i by the conditions $z_i = 1$ and $z_i = 0$ respectively.

First let H: Clauses $\rightarrow$ Terms be defined as follows:

$H(c_i) = f(z_1, z_2, z_3, u_{i1}, u_{i2})$ if $c_i = \{z_1, z_2, z_3\}$ or if $c_i = \{\overline{z_1}, \overline{z_2}, \overline{z_3}\}$

and let G: Clauses $\rightarrow$ Terms be defined as:

$$G(c_i) = f(1, 1, 1, 0, 0) \text{ if } c_i = \{z_1, z_2, z_3\}$$
$$= f(0, 0, 0, 1, 1) \text{ if } c_i = \{\overline{z_1}, \overline{z_2}, \overline{z_3}\}$$

For example if $c_2 = \{p, q, r\}$ then $H(c_2) = f(p, q, r, u_{21}, u_{22})$ and $G(c_2) = f(1, 1, 1, 0, 0)$. Clearly, for each i, c_i is satisfiable if and only if $H(c_i)$ can be matched with $G(c_i)$: if c_i is all-positive, then at least one of its variables will have to be matched with 1, and if c_i is all-negative then at least one of its variables will have to be matched with 0. Now let $t_1 = g(s_1, s_2, \cdots, s_n)$ where $s_i = H(c_i)$ and $t_2 = g(s_1', s_2', \cdots, s_n')$ where $s_i' = G(c_i)$. It is obvious that these terms can be constructed in polynomial time with respect to the input.

The reader can now easily see that t_1 can be ac-matched with t_2 if and only if C is satisfiable. □

The above construction also shows that the problem remains NP-complete even if there is only one ac-operator and at least three operators that are neither commutative nor associative. *

Since associative-commutative matching is a special case of associative-commutative unification as pointed out by Peterson and Stickel (1981), associative-commutative unification is NP-hard. However, it will be interesting to develop a complexity bound for associative-commutative unification.

5. Restricted Associative-Commutative Matching

In this section, we show that a restricted associative-commutative matching problem, namely DO-ACM, can be done in polynomial time. The idea behind the algorithm is this: if two terms, say t_1 and t_2, having no variable at all in common can be matched with s_1 and s_2 respectively, then a substitution can be found that does both matches "simultaneously." This substitution is merely the **union** of the two former substitutions since no conflicts are possible.

Consider two flattened terms $t = f(t_1, t_2, \cdots, t_n)$, $s = f(s_{n+1}, \cdots, s_{n+m})$, where f is an ac-operator. It should not be hard to see that if $n > m$ then no match is possible. Consider the case when $m > n$. Clearly, at least one of the t_i's must be a variable. Without loss of generality, let $t_1, \ldots, t_k$ $(k \leq n)$ be the non-variable terms. Now all we have to do is this: for each non-variable t_i, find an s_j that it can be matched with, making sure that no two t_i's are matched with the same s_j. In other words, what we need is an **injection** π from $\{1, \cdots, k\}$ into $\{n+1, \cdots, m\}$ such that t_i can be matched with $s_{\pi(i)}$. This can be found recursively by the following steps:

* The NP-completeness of associative-commutative matching was also shown independently by Ashok Chandra and Paris Kanellakis by reducing the bin-packing problem to it.

a. For every t_i find all s_j's that it can be matched with. This can be done by an exhaustive search.

b. Form a k-by-m undirected bipartite graph G with nodes corresponding to each t_i and s_j such that there is an edge between nodes t_i and s_j if and only if t_i can be matched with s_j.

c. Check to see if there is a matching of size k.

The case when $n = m$ is similar.

The following example will illustrate the method: let $t = f\,(g\,(x\,,\,a\,),\,g\,(b\,,\,b\,,\,y\,))$ and $s = f\,(g\,(a\,,\,b\,,\,b\,),\,g\,(a\,,\,a\,))$ where f and g are ac-operators. We get the graph G (see Figure 1)

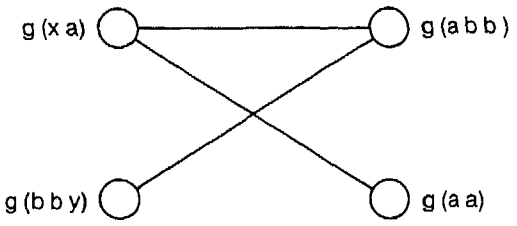

Figure 1.

and the maximum matching (of size 2) is the one that matches $g\,(x\,,\,a\,)$ with $g\,(a\,,\,a\,)$ and $g\,(b\,,\,b\,,\,y\,)$ with $g\,(a\,,\,b\,,\,b\,)$. Thus the substitution is $\{x \leftarrow a\,,\,y \leftarrow b\,\}$.

Step (c) can be done in time $O\,(m^3)$ since one can find a maximum matching of a bipartite graph $G = <V\,,E>$ in time $O\,(\,|\,V\,|\,*\,|\,E\,|\,)$ (Papadimitriou and Steiglitz, 1982). Thus given the graph G we can determine if there exists a θ such that $\theta(t) \underset{ac}{=} s$ in $O\,(m^3)$ time.

In order to determine the graph G and to determine whether a suitable substitution exists we can write a recursive algorithm that first at-

tempts to find the set of edges of G by calling itself with inputs t_i and s_j for each i and j. We now leave it as an easy exercise to the reader to show that the overall running time of the algorithm is $O(|s|^3 \cdot |t|)$.

The case when the top-level function symbol of t and s is neither commutative nor associative is trivial. Thus

Theorem 2: DO-ACM can be solved in polynomial time.

Corollary 3: ACEQ can be done in polynomial time.
Proof: This is easy to see since ACEQ is equivalent to DO-ACM if we consider all the variables in the two terms to be constants. □

Associative-commutative matching, however, remains NP-complete even when no variable in a pattern has more than three occurrences (Kapur and Narendran, 1986). The complexity of the case when a pattern has at most two occurrences of variables is still an open question.

6. Associative Matching (AM)

Instance: A set of function symbols F some of which may be associative, a set of variables V and two terms t_1, t_2 from $T(F, V)$.

Question: Does there exists a θ such that $\theta(t_1) \underset{A}{=} t_2$?

Theorem 4: AM is NP-complete.
Sketch of the proof: It is easy to see that AM is in NP. The construction employed in showing that AM is NP-hard is given below.

Let $C = \{c_1, c_2, \cdots c_m\}$ be an instance of 3SAT over the boolean variables $x_1, \ldots, x_n$. Take $F = \{f, a, h\}$ where f is an associative function symbol, a is a nullary (constant) symbol and h is an $(n+m)$-ary function symbol.

Let $V = \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_n\} \cup \{u_1, \ldots, u_m\}$, where each y_i acts the role of $\overline{x_i}$ and the u_i's are dummy symbols. The truth and falsity of a boolean variable x_i is simulated by the conditions $x_i = a$ and $x_i = f(a, a)$ respectively.

Define, for $1 \leq i \leq n$,

$$s_i = f(x, y) \text{ and}$$
$$t_i = f(a, f(a, a)) = f(a, a, a).$$

(Note that if s_i has to be matched with t_i, either x_i or y_i must be made equal to a and not both.) For each clause c_j, we do the following: let x_1, x_2 and x_3 be the variables in c_j. Then we set

$$p_j = f(z_1, z_2, z_3, u_j) \text{ and}$$
$$q_j = f(a, a, a, a, a, a).$$

where $z_i = x_i$ if the literal x_i appears in c_j and $z_i = y_i$ if the literal $\overline{x_i}$ appears in c_j, for $i = 1,2,3$. Here again, note that to match p_j with q_j at least one of the z_i's must be set equal to a.

Finally set $s = h(s_1, \ldots, s_n, p_1, \ldots, p_m)$ and $t = h(t_1, \ldots, t_n, q_1, \ldots, q_m)$. The reader can easily verify that s can be associative-matched with t if and only if C is satisfiable. □

Thus the problem remains NP-complete even if there is only one associative operator and at least two non-associative operators. The result also implies that the associative unification problem is NP-hard. Related results are claimed in Iwama (1982).

7. Commutative matching (CM)

Instance: A set of function symbols F some of which may be commutative, a set of variables V and two terms t_1, t_2 from $T(F, V)$.

Question: Does there exists a θ such that $\theta(t_1) \underset{C}{=} t_2$?

Theorem 5: CM is NP-complete.

Sketch of the proof: The construction is as follows:

Let $C = \{c_1, c_2, \cdots c_m\}$ be an instance of 3SAT over the boolean variables $x_1, \ldots, x_n$. Take $F = \{f, g, a, h, 0, 1\}$ where f is a binary commutative function symbol, g is a ternary function symbol, a, 0 and 1 are nullary (constant) symbols and h is an m-ary function symbol.

Let $V = \{x_1, \ldots, x_n\} \cup \{u_{ij} \mid 1 \leq i \leq m, 1 \leq j \leq 3\}$. The truth and falsity of a boolean variable x_i is simulated by the conditions $x_i = 1$ and $x_i = 0$ respectively.

For each clause c_j, we do the following: let x_1, x_2 and x_3 be the variables in c_j. There are exactly 7 sets of truth-value-assignments that make the clause c_j true. Let $q_1, \ldots, q_7$ be 7 (distinct) terms that 'represent' these assignments; for $i = 1, \ldots, 7$, define $q_i = g(b_1, b_2, b_3)$ where $b_i \in \{1,0\}$ and the assignment (b_1, b_2, b_3) satisfies c_j. Let

$$s_j = f(f(f(g(x_1, x_2, x_3), u_{j1}), u_{j2}), u_{j3}) \text{ and}$$
$$t_j = f(f(f(q_1, q_2), f(q_3, q_4)), f(f(q_5, q_6), f(q_7, a))).$$

(See Figure 2)

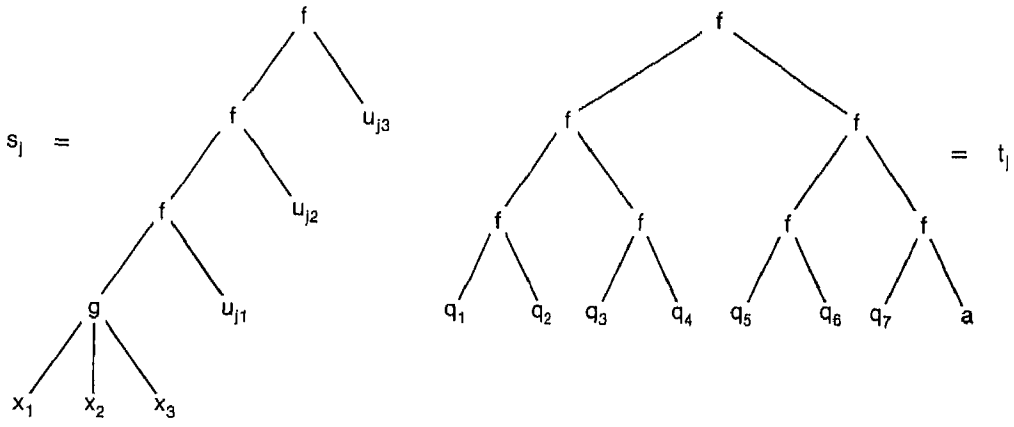

Figure 2.

Finally define $s = h(s_1, \cdots, s_m)$ and $t = h(t_1, \cdots, t_m)$. It can be shown that s can be commutative-matched with t if and only if C is satisfiable. □

As in the case of the two previous NP-completeness results this proof also shows that CM remains NP-complete even if there is only one commutative operator and at least five non-commutative operators. The result is also interesting in another respect. The commutative unification problem is known to be NP-complete (see p. 252 in Garey and Johnson (1979) where this result is attributed to Ravi Sethi). For

commutative operators, the matching problem and unification problem thus surprisingly turn out to be of the same complexity. This is similar to the case of unification and matching problems over a free theory where the complexity is linear for both problems.

We subsequently proved that the set-matching and set-unification problems, which are, respectively, special cases of associative-commutative-idempotent matching and unification (with the associative-commutative-idempotent function appearing only as the outermost function in a pattern), are NP-complete (Kapur and Narendran, 1986). This has led us to show that even if an associative-commutative (associative or commutative) operator is idempotent and/or has an identity element, the matching problem remains NP-hard.

In an earlier version of this paper (Benanav et al, 1985), we had conjectured that associative-commutative unification problem is NP-complete. Recently, Kapur and Narendran (1986) have been able to settle this conjecture positively by showing that the check for associative-commutative unification can be performed in NP time.

The NP-completeness of associative-commutative matching problem (ACM) was established during the course of discussions with Snorri Agnarsson who contributed ideas for simplifying the proof. We also thank the referees for their helpful comments and suggestions. The first two authors were partially supported by the National Science Foundation grant MCS-82-11621.

8. References

Benanav, D., Kapur, D. and Narendran, P. (1985). Complexity of Matching Problems. In: Proceedings of *Rewriting Techniques and Applications*, Dijon, France. LNCS 202, Berlin: Springer Verlag.

Dwork, C., Kanellakis, P. and Stockmeyer, L. (1986). Parallel Algorithms for Term Matching. In: Proceedings of *8th Conference on Automated Deduction (CADE-86)*, Oxford, U.K. LNCS 230, Berlin: Springer Verlag.

Garey, M.R. and Johnson, D.S. (1979). *Computers and Intractability*, W.H. Freeman.

Hoffman, C.M. and O'Donnell, M.J. (1982). Pattern Matching in Trees. *Journal of Assoc. of Comp. Mach.* **29**, 68-95.

Hullot, J.M. (1979). Associative-Commutative Pattern Matching. *Fifth International Joint Conference on Artificial Intelligence*, Tokyo, Japan.

Iwama, K. (1982). On Equations Including String Variables. In: *Proc. 23rd Ann. Symp. on Foundations of Computer Science*, 226-235.

Kapur, D. and Narendran, P. (1986). NP-completeness of the Set Unification and Matching Problems. In: Proceedings of *8th Conference on Automated Deduction (CADE-86)*, Oxford, U.K. LNCS 230, Berlin: Springer Verlag.

Kapur, D. and Narendran, P. (1986). Associative-commutative unification is NP-complete. Unpublished manuscript, General Electric Corporate Research and Development Center, Schenectady, NY.

Kapur, D. and Narendran, P. (1987). Matching, Unification, and Complexity. To appear as a General Electric Corporate Research and Development Report.

Kapur, D., Sivakumar, G. and Zhang, H. (1986). RRL: A Rewrite Rule Laboratory. In: Proceedings of *8th Conference on Automated Deduction (CADE-86)*, Oxford, U.K. LNCS 230, Berlin: Springer Verlag.

Knuth, D.E. and Bendix. P.B. (1970). Simple Word Problems in Universal Algebras. in *Computational Problems in Abstract Algebras* (ed. J. Leech), Pergamon Press, 263-297.

Lankford, D.S. (1975). Canonical Inference. Report ATP-32, Dept. of Mathematics and Computer Sciences, Univ. of Texas, Austin, Texas.

Lankford, D.S., and Ballantyne A.M. (1977). Decision Procedures for Simple Equational Theories with Commutative-Associative Axioms: Complete Sets of Commutative-Associative Reductions. Memo ATP-39, Dept. of Mathematics and Computer Sciences, Univ. of Texas, Austin, Texas.

Lankford, D.S., and Ballantyne, A.M. (1979). The Refutation Completeness of Blocked Permutative Narrowing and Resolution. In: *4th Conference on Automated Deduction (CADE).*

Overmars, M.H. and Van Leeuwen, J. (1979). Rapid Subtree Identification Revisited. Technical Report RUU-CS-79-3, Dept. of Computer Science, University of Utrecht, Utrecht.

Papadimitriou, C.H., and Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity,* Prentice-Hall.

Peterson, G.E., and Stickel, M.E. (1981). Complete Sets of Reductions for Some Equational Theories. *Journal of Assoc. of Comp. Mach.* **28/2**, 233-264.

Plotkin, G. (1972). Building in Equational Theories. *Machine Intelligence* **7** (eds. Meltzer and Michie), 73-90.

Slagle, J. (1974). Automated Theorem Proving with Simplifiers, Commutativity and Associativity. *Journal of Assoc. of Comp. Mach.* **21**, 622-642.

Steyaert, J.-M. and Flajolet, P. (1983). Patterns and Pattern-Matching in Trees: An Analysis. *Information and Control.* **28/1-3**, 19-58.

Stickel, M.E. (1980). A Unification Algorithm for Associative-Commutative Functions. *Journal of Assoc. of Comp. Mach.* **28**, 423-434.